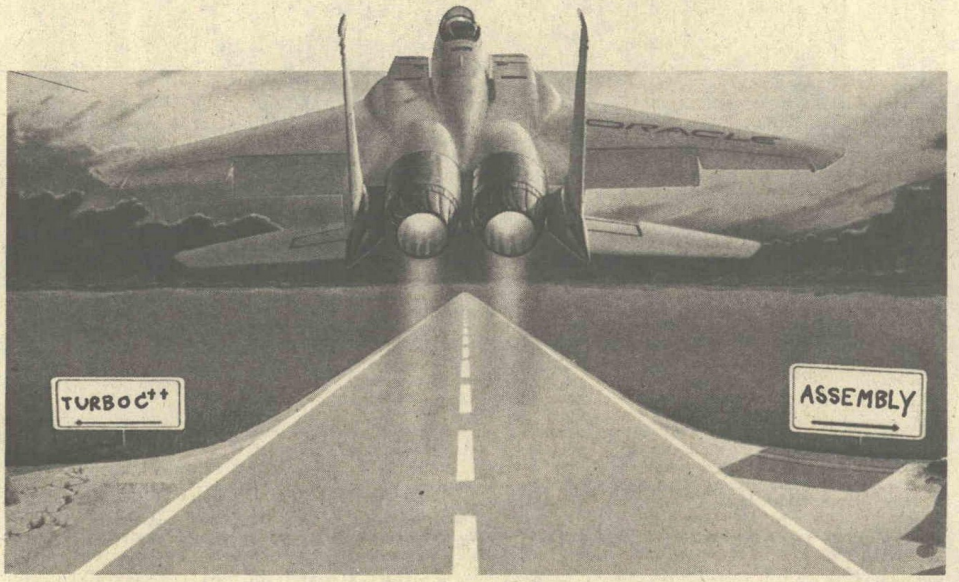
สวาท โพธิปัสสา

SOFT MAIL

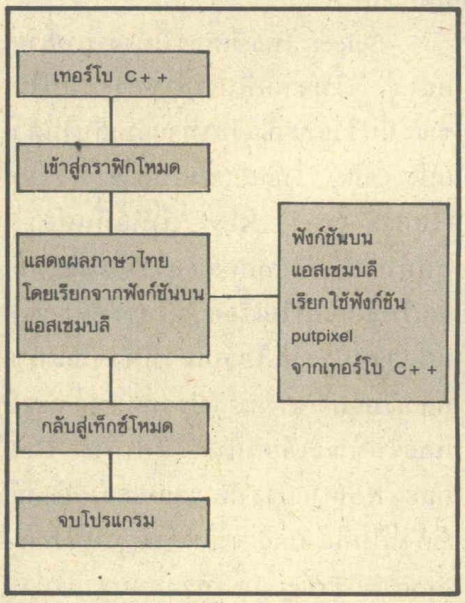
ไทยฟอนต์ ในเทอร์โบ C++ กับแอสเซมบลี

หนึ่งบวกหนึ่งเป็นสอง แล้วเทอร์โบ C++ กับแอสเซมบลีมารวมกันจะเกิดอะไรขึ้น กับภาษาไทย บนไมโครคอมพิวเตอร์



เนื่องจากผู้เขียนได้พยายามเขียนโปรแกรมการแสดงภาษาไทยในกราฟิกโหมด เพื่อใช้งานด้านการเรียนการสอน โดยตอนแรกทดลองใช้เทอร์โบปาสคาลเขียนโปรแกรมสร้างฟอนต์ใช้รหัสเกษตร (โปรแกรมสร้าง, แก้ไขฟอนต์อยู่ที่ไฟล์ FONTEDIT.EXE ในแผ่นดิสก์ที่ผู้เขียนเตรียมไว้ให้แล้ว) แล้วทำการเขียนโปรแกรมโหลดฟอนต์ขึ้นมาแสดงบนจอในโหมดกราฟิกโดยใช้เทอร์โบปาสคาลเขียนเป็นส่วนใหญ่ แต่ประสบปัญหาทางด้านความเร็วในการแสดงผลบนจอ ดังนั้นจึงหันมาใช้แอสเซมบลีเข้าช่วยแต่การเชื่อมโยงระหว่างแอสเซมบลีกับเทอร์โบปาสคาลทำให้เกิดปัญหามาก ตัวอย่างเช่น เราจะใช้คำสั่ง Call จากแอสเซมบลีเรียกใช้คำสั่งที่เป็นมาตรฐานของเทอร์โบปาสคาลไม่ได้ ดังนั้นผู้เขียนจึงหันมาใช้เทอร์โบ C++ และเทอร์โบแอสเซมบลี เวอร์ชัน 1.0 เพื่อใช้ในงานนี้และที่สำคัญนอกจากทำงานได้เร็วขึ้นแล้วยังสามารถแสดงผลฟอนต์ภาษาไทยบนจอแสดงผลได้เกือบทุกชนิดอีกด้วย

ผู้อ่านบางท่านอาจจะสงสัยว่า "ทำไมต้องมีการเขียนแอสเซมบลีเพื่อเรียกใช้คำสั่งบนเทอร์โบปาสคาลหรือเทอร์โบ C++ ทำไมไม่เขียนแอสเซมบลีล้วน ๆ" สาเหตุเนื่องจากเทอร์โบ C++ และเทอร์โบปาสคาลเองได้มีการจัดเตรียมคำสั่งและฟังก์ชันต่าง ๆ ทางกราฟิกไว้อยู่ในไลบรารีและยูนิต ซึ่งถือว่าเป็นมาตรฐาน เพราะสามารถนำไปใช้งานในด้านการแสดงผลทางกราฟิกได้บนการ์ดหลายชนิด เช่น เฮอร์คิวลิส, อีจีเอ, วีจีเอ, ซีจีเอ เป็นต้น หัวใจสำคัญในการเขียนบทความนี้คือการใช้แอสเซมบลีเรียกใช้ฟังก์ชันการแสดงผลบนจอในโหมดกราฟิกของเทอร์โบ C++ ซึ่งจะใช้ฟังก์ชันการสร้างจุด putpixel (int x, int y, int color) จากนั้นเราจะสร้างฟังก์ชันการแสดงผลภาษาไทยโดยใช้แอสเซมบลีและใช้เทอร์โบ C++ เรียกใช้อีกทีหนึ่ง ดังรูปที่ 1



รูปที่ 1 วิธีการแสดงฟอนต์ไทยในโหมดกราฟิก

รายละเอียดต่าง ๆ ในการเขียนโปรแกรมลิงค์ระหว่างเทอร์โบซีกับแอสเซมบลีอ่านได้จากมุมภาษาซีในไมโครฉบับที่ 51 ของ อ.จเร เลิศสุดวิชัย

วิธีการเรียกใช้ฟังก์ชัน PUTPIXEL ของเทอร์โบ C++

เราสามารถใช้โปรแกรมภาษาแอสเซมบลีเรียกใช้ฟังก์ชันจากเทอร์โบ C++ ได้ทุกฟังก์ชัน แต่ในที่นี้จะขอยืดฟังก์ชัน __putpixel เพื่อใช้อธิบายในการแสดงฟอนต์ สรุปได้ดังนี้ (ดูโปรแกรมในรูปที่ 2 ประกอบ)

1. ต้องอยู่ในกราฟิกโหมดมีความละเอียดสูง

2. ต้องกำหนดโมเดล (model) ให้ตรงกัน ในที่นี้ใช้โมเดลสมอร์ (Model small)

3. ประกาศให้ฟังก์ชัน putpixel เป็นเอ็กซ์เทอร์แนลและมีการ call แบบ far

4. การเรียกใช้ฟังก์ชัน putpixel นี้จะทำลายรีจิสเตอร์ AX, BX, CX และ DX ฉะนั้นถ้ามีการใช้รีจิสเตอร์เหล่านี้อยู่ในโปรแกรมก่อนแล้วการเรียกใช้ฟังก์ชัน putpixel จะต้องมีการรักษาค่ารีจิสเตอร์เหล่านี้โดยอาศัยการ push ลงสแต็กหลังจากออกจากฟังก์ชัน putpixel จึง pop ค่าออกจากสแต็ก

5. ส่งผ่านค่าอาร์กิวเมนต์ทางสแต็ก โดย push ค่าทางขวาไปซ้ายเสมอ ค่าอาร์กิวเมนต์ของฟังก์ชัน putpixel (int x, int y, int color) ได้แก่ ค่า x, y และ color จะต้องเอาค่า color ลงสแต็กก่อนแล้วตามด้วยค่า y และ x ตามลำดับจากขวามาซ้าย

```
;----- Define Segment and Model Same turbo C++ -------

            DOSSEG
            .MODEL  SMALL
            .CODE
            Extrn  _Putpixel:far
            Public _TestLine

; function  call  by  TURBO C++
; Void  TestLine (Void)
;       Line Start from co-ordinate (10,100) to (200,100)
;-------------------------------------------------------------
_TestLine       PROC    near
                MOV     AX,01H      ;Get color blue
                MOV     CX,10       ;Get Start X co-ordinate
                MOV     DX,100      ;Get Y co-ordinate
Line1:
                PUSH    AX          ; Color
                PUSH    DX          ; Y
                PUSH    CX          ; X
                Call    far ptr _putpixel  ; Geat dot
                POP     CX          ; Return Variable or Stack
                POP     DX          ; Same Add SP,6
                POP     AX
                INC     CX          ; x = x +1
                CMP     CX,201      ; End of Point
                JNZ     Line1
                RET
_Test Line      Endp
;-------------------------------------------------------------
                END
```

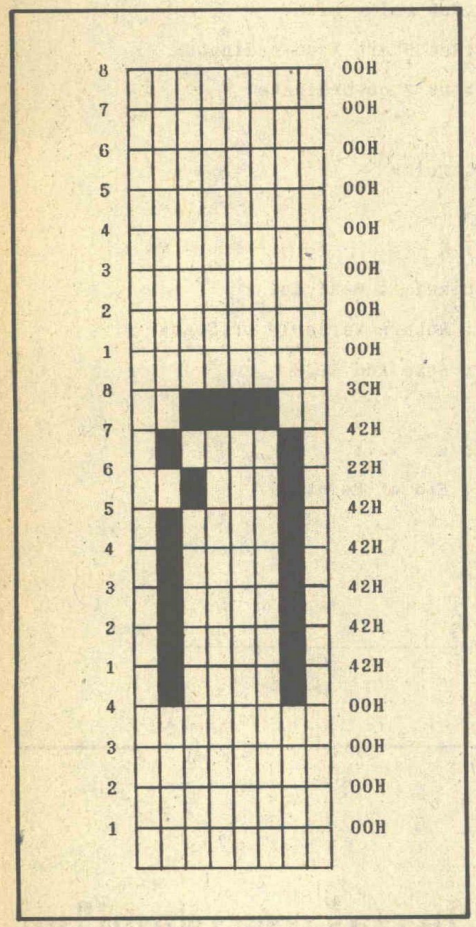
รูปที่ 2 แสดงตัวอย่างการเรียกใช้ฟังก์ชัน putpixel

จากรูปที่ 2 แสดงตัวอย่างการเรียกใช้ฟังก์ชัน putpixel บนแอสเซมบลีซึ่งจะเป็นฟังก์ชัน TestLine ซึ่งสามารถเรียกได้จากเทอร์โบ C++ ในการลากเส้นทางแนวนอน จากจุด (10, 100) ไปยังจุด (200,100) ด้วยสีน้ำเงิน

โครงสร้างของฟอนต์ไทย

ในบทความนี้เราจะใช้รูปแบบของฟอนต์ที่สร้างจากโปรแกรมฟอนต์ Edit. EXE ซึ่งเก็บอยู่ในรหัสเกษตร โดยฟอนต์ไฟล์ที่ได้จะถูกเก็บลงไฟล์ 2

ไฟล์ คือ THAIFONT.CHR และ THAIIFNT.CHR ฟอนต์ที่ได้จาก THAIFONT.CHR เป็นฟอนต์แบบธรรมดาส่วนอีกไฟล์เป็นแบบตัวเอียง ซึ่งการแสดงผลแต่ละตัวอักษรจะใช้แสดงแบบดอตเมตริกซ์ขนาด 8×20 (ดูรูปที่ 3) ดังนั้นเมื่อเก็บข้อมูลลงไฟล์จำนวนตัวอักษรละ 8×20 บิต หรือ 20 ไบต์ และจำนวน 256 ตัวอักษรจะใช้เนื้อที่ทั้งไฟล์ 256×20 ไบต์ = 5120 ไบต์



รูปที่ 3 แสดงโครงสร้างของตัวอักษร ก ซึ่งจะใช้ข้อมูล 20 ไบต์ (ทางขวามือ) เป็นตัวกำหนดรูปร่าง โดยในแต่ละไบต์แบ่งได้เป็น 8 บิต ในแต่ละบิตถ้าบิตใดเป็น '0' หมายถึงไม่เป็นจุด (dot) ถ้าบิตใดเป็น '1' จะเกิดจุดขึ้น

การกำหนดตัวแปรเพื่อการแสดงผลของฟอนต์

เพื่อต้องการให้การแสดงฟอนต์ได้หลายขนาด หลายสี หลายฟอนต์ ฯลฯ จึงได้มีการกำหนดค่าตัวแปรขึ้นเพื่อใช้ในฟังก์ชันที่สร้างขึ้นด้วยแอสเซมบลีดังรูปที่ 4

ค่าต่าง ๆ ที่กำหนดไว้นี้สามารถเปลี่ยนแปลงและทำการแก้ไขได้ (ยกเว้นแพตเทอร์น) ผู้เขียนได้เตรียมฟังก์ชันที่เขียนด้วยภาษาแอสเซมบลีซึ่งสามารถเรียกใช้ได้จากเทอร์โบ C++ ทุกครั้ง และจะได้อธิบายหน้าที่ของแต่ละฟังก์ชันในภายหลัง

หลักการแสดงฟอนต์บนจอ

หลักการง่าย ๆ คือจะนำค่าในรหัสแอสกีที่จะแสดงผลมาคำนวณหาตำแหน่งที่เก็บฟอนต์สามารถหาได้จาก

แอดเดรสเริ่มต้นของฟอนต์แต่ละตัวอักษร (ASCII) = แอดเดรสเริ่มต้นของฟอนต์ทั้งหมด + (ASCII CODE × 8)

ตัวอย่างเช่น เราต้องการทราบที่เก็บแอดเดรสเริ่มต้นของ 'ก' แบบตัวเอียงอยู่ที่ใด โดยรหัสแอสกีของ 'ก' มีค่าเท่ากับ 161

แอดเดรสเริ่มต้นของฟอนต์ 'ก' แบบตัวเอียง = IByte + (161×8)

จากนั้นก็ทำการอ่านข้อมูลจากตำแหน่งที่คำนวณได้มาทีละไบต์ แล้วทำการหมุนข้อมูลไปทางขวามือ ผ่าน cary flag จากนี้ตรวจสอบว่า cary flag เป็น '1' หรือไม่ ถ้าเป็น '1' ก็ให้เรียกฟังก์ชัน putpixel เพื่อสร้างจุดบนจอต่อไป เมื่อเราหมุนข้อมูลแต่ละไบต์ครบ 8 บิตและครบจำนวน 20 ไบต์และสร้างจุดบนจอให้ตรงตามตำแหน่งก็จะเกิดฟอนต์บนจอได้ส่วนการสร้างขนาดของตัวอักษรนั้นจะใช้ procTBar เพื่อสร้างจุดตามขนาดความกว้าง (XWidth) และความสูง (YWidth) ของจุดที่อ่านได้จากตัวแปร XWidth และ YWidth ตามลำดับ และสร้างสีตามตัวแปร Thaicolor

ฟังก์ชันของแอสเซมบลีที่เกี่ยวกับฟอนต์

เพื่อให้เกิดความรวดเร็วในการแสดงผลของฟอนต์ ฟังก์ชันที่เกี่ยวกับฟอนต์ต่าง ๆ ได้เขียนเป็นโปรแกรมภาษาแอสเซมบลีทั้งหมด แล้วสามารถเรียกใช้ฟังก์ชันต่าง ๆ เหล่านี้ได้จากเทอร์โบ C++ ลักษณะการเขียนเป็นฟังก์ชันแยกไปต่างหากทำให้สะดวก

```
ThaiColor   DW  1                   ; สีของฟอนต์ ค่าเริ่มต้นคือสีน้ำเงิน
XWitbl      DW  1,1,.....8          ; ตารางขนาดความกว้างของจุด ในการสร้างฟอนต์
YWitbl      DW  1,1,.....8          ; ตารางขนาดความสูงของจุด ในการสร้างฟอนต์
XWidth      DW  1                   ; ความกว้างของจุดที่จะนำไปใช้งาน
YWidth      DW  1                   ; ความสูงของจุดที่จะนำไปใช้งาน
TByte       DB  20×256 dup (0)      ; ที่เก็บแพตเทอร์นของฟอนต์ตัวธรรมดา
IByte       DB  20×256 dup (0)      ; ที่เก็บแพตเทอร์นของฟอนต์ตัวเอียง
Tfont       DB  00000000            ; ใช้บ่งบอกการเลือกฟอนต์
                                    ; และเพื่อเก็บไว้สำหรับแอดทริบิวต์ในอนาคต
```

รูปที่ 4 การกำหนดตัวแปรเพื่อใช้ในแอสเซมบลี

ในการแก้ไขปรับปรุงรูปแบบการแสดงฟอนต์แบบต่าง ๆ ในอนาคตได้และการเรียกใช้จากเทอร์โบ C++ ของฟังก์ชันต่าง ๆ มีดังนี้

1. ฟังก์ชัน loadThaifont ( ) ทำหน้าที่โหลดข้อมูลฟอนต์ธรรมดาจากไฟล์ THAIFONT.CHR เข้าไปยังหน่วยความจำเริ่มต้นที่ TBytes แล้วยังให้ค่าส่งกลับเป็น 0 ถ้าโหลดข้อมูลสำเร็จ แต่ถ้าไม่เป็น 0 หมายถึงมีข้อผิดพลาดระหว่างการโหลดฟอนต์ไฟล์

2. ฟังก์ชัน load ThaiIfont ( ) ทำหน้าที่โหลดข้อมูลฟอนต์ตัวเอียง (ITALIC) จากไฟล์ THAIIFNT.CHR เข้าไปยังหน่วยความจำเริ่มต้นที่ IBytes แล้วส่งค่ากลับเป็น 0 ถ้าโหลดข้อมูลสำเร็จ แต่ถ้าไม่เป็น 0 หมายถึงมีข้อผิดพลาดระหว่างการโหลดฟอนต์ไฟล์

3. ฟังก์ชัน Tprints (int X, int Y, char * Strings) จะพิมพ์ข้อความ Strings ตามฟอนต์ที่เลือกในปัจจุบันลงบนจอแสดงผล โดยเริ่มที่ตำแหน่ง (X,Y) ฟังก์ชันนี้ใช้ได้ในขณะอยู่ในกราฟิกโหมดเท่านั้น

4. ฟังก์ชัน Tprintc (int X, int Y, Char ch) จะพิมพ์ตัวอักษรหนึ่งตัว (ch) ลงบนจอแสดงผล ณ ตำแหน่ง (X,Y) ใช้ได้ในขณะอยู่ในกราฟิกโหมดเท่านั้น

5. ฟังก์ชัน selectThaifont (int attr) เป็นการเลือกฟอนต์แสดงผล โดยถ้าให้ attr เป็น 0 จะเป็นการแสดงฟอนต์ปกติ แต่ถ้า attr เป็น 1 จะเป็นการแสดงฟอนต์แบบตัวเอียง

6. ฟังก์ชัน setuserThaiwidth (int XWidth, int YWidth) เป็นฟังก์ชันที่ใช้ในการกำหนดขนาดของจุดที่นำมาสร้างฟอนต์ใน proc TBar โดย XWidth คือความกว้างจุด YWidth คือความสูงของจุด

7. ฟังก์ชัน selectThaisize (int Tsize) ฟังก์ชันนี้ใช้ในการเลือกขนาดของตัวอักษร ซึ่ง Tsize มีค่าตั้งแต่ 0 ถึง 16

8. ฟังก์ชัน setcelThaicolor (int color) เมื่อเราจะกำหนดสีให้แก่ตัวอักษรให้เรียกใช้ฟังก์ชันนี้ โดยค่าสี (color) ต่าง ๆ ได้ถูกกำหนด (define) ค่าไว้ในไฟล์ graphics.h มีประมาณ 16 สีขึ้นอยู่กับจอแสดงผล เช่นถ้าเป็นการ์ดโมโนโครมจะมี 2 สี คือ สีดำและสีน้ำเงิน

โปรแกรมเทอร์โบ C++ ที่เรียกใช้ฟังก์ชันบนแอสเซมบลี

โดยฟังก์ชัน Initialize ( ) ใช้ในการเปลี่ยนเท็กซ์โหมดไปเป็นกราฟิกโหมดโดยมีการตรวจสอบการ์ดแสดงผลแบบอัตโนมัติและฟังก์ชัน Initialize ( ) นี้จะเรียกไฟล์ที่มีนามสกุลเป็น .BGI โดยชื่อของไฟล์ขึ้นอยู่กับชนิดของการ์ดแสดงผล เช่น HERC.BGI, EGAVGA.BGI, CGA.BGI เป็นต้น (สามารถรวมไฟล์ .BGI เข้ากับ graphics.LIB ได้ แต่จะกล่าวถึงในคราวหน้า กรุณาอดใจรอ) ส่วนการกลับสู่เท็กซ์โหมด (text-mode) ใช้ฟังก์ชัน closegraph( )

ในฟังก์ชัน main( ) จะพบว่าก่อนเริ่มต้นทำงานต้องมีการโหลดฟอนต์ภาษาไทยเข้าไปในหน่วยความจำก่อนที่จะมีการแสดงฟอนต์ ตัวอย่างการแสดงฟอนต์อยู่ที่ฟังก์ชัน ShowThaiDemo( ) ภาษาไทยนั้นจะต้องใช้รหัสเกษตรหรืออาจจะใช้การ์ดภาษาไทยเข้าช่วย แต่ถ้าไม่มีการ์ดภาษาไทยผู้เขียนขอแนะนำให้ใช้เทอร์โบซีเขียนโปรแกรมในส่วนอื่น ๆ ก่อน และในฟังก์ชันที่

จะแสดงผลภาษาไทย เช่น Tprints (' ') ขอให้เว้นไว้ จากนั้นเก็บโปรแกรมลงแผ่นดิสค์ เสร็จแล้วเรียกใช้โปรแกรม CU WRITER ขึ้นมาเพิ่มเติมภาษาไทยภายหลัง (แต่อย่าลืมตั้งค่าตัวเลือกใน CU WRITER จะต้องเป็นรหัสเกษตร) ถ้าเพิ่มเติมส่วนภาษาไทยเสร็จแล้วก็เก็บโปรแกรมลงแผ่นดิสค์และเพื่อความแน่ใจควรใช้โปรแกรมเทอร์โบซีแก้ไขตบแต่งอีกครั้ง เพราะว่ารหัสบางตัวของ CU WRITER ไม่ตรงกับเทอร์โบซี (TC)

การคอมพายล์และลิงค์

การคอมพายล์และลิงค์โปรแกรมที่ได้จากเทอร์โบ C++ เข้ากับแอสเซมบลีนั้นสามารถทำได้โดยง่ายในฮาร์ดดิสค์ (เพราะผู้เขียนใช้ฮาร์ดดิสค์ตลอดสำหรับบทความนี้) เริ่มต้นควรใส่พาธ (PATH) เพื่อให้โปรแกรมสามารถค้นหาไฟล์ต่าง ๆ มาเอ็กซีคิวต์ (Execute) ได้ ซึ่งอาจจะใส่ไว้ในไฟล์ AUTOEXEC.BAT หรือพิมพ์บนดอสพร้อมเลยก็ได้ ตัวอย่าง เช่น

```
PATH = D:\TC\BIN;D:\ASSEMBLY
\TASM
```

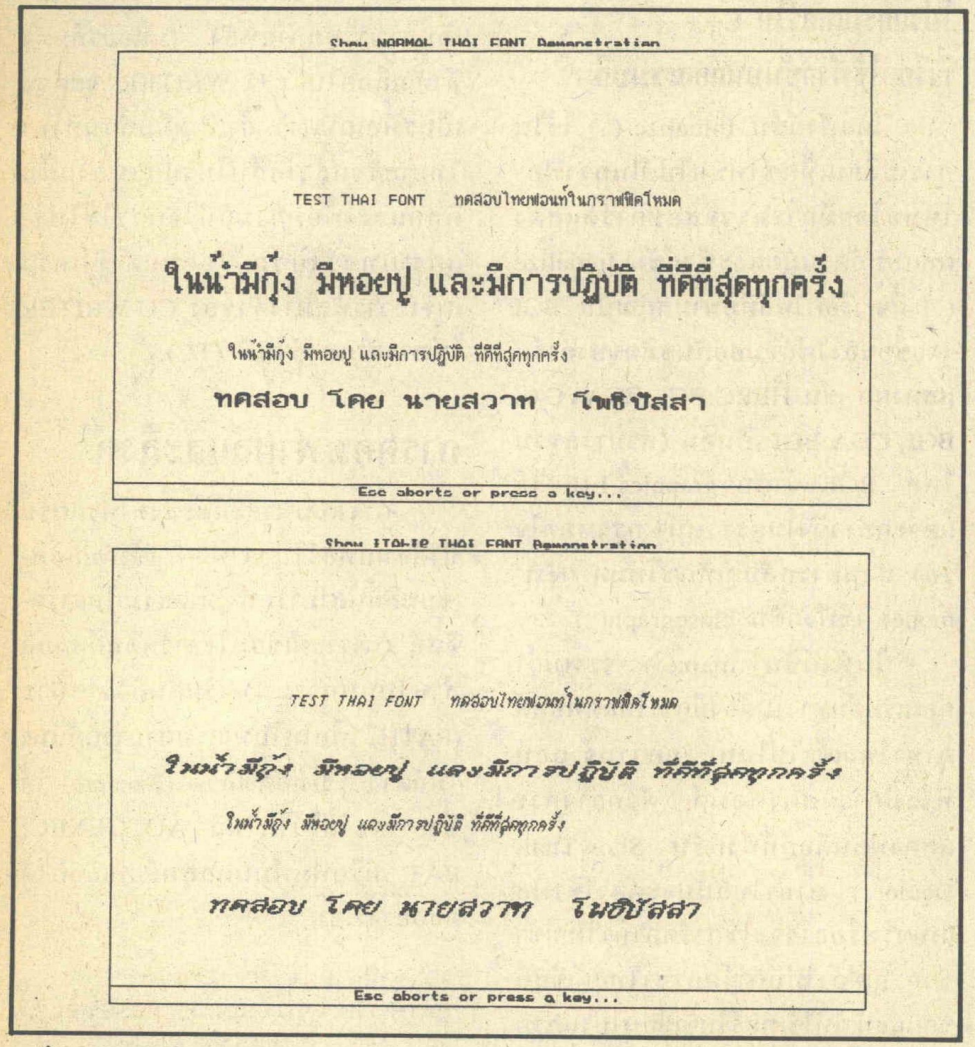
```
D:\TC\OUTPUT>tcc -ms thai.c thaifont.asm graphics.lib
Turbo C++  Version 1.00 Copyright (c) 1990 Borland International
thai.c:
thaifont.asm:
Turbo Assembler  Version 1.0  Copyright (c) 1988 by Borland International

Assembling file:   THAIFONT.ASM
Error messages:    None
Warning messages:  None
Remaining memory:  319k

Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International

        Available memory 323312
```

รูปที่ 5 แสดงผลการคอมพายล์



รูปที่ 6 ผลการรันโปรแกรม THAI.EXE

ในที่นี้ เราเก็บไฟล์ต้นฉบับ (Soure file) จากเทอร์โบซี C++ ไว้ที่ไฟล์ชื่อ THAI.C และแอสเซมบลีเก็บไว้ที่ไฟล์ชื่อ THAIFONT.ASM การคอมพายล์จะใช้คำสั่ง (Command line) จากเทอร์โบ C++ ดังนี้

```
TCC -ms THAI.C THAIFONT.ASM
GRAPHICS.LIB
```

ส่วนผลการคอมพายล์แสดงไว้ในรูปที่ 5 โดยผลลัพธ์จะได้ไฟล์ THAI.EXE และผลการรันโปรแกรม THAI.EXE แสดงดังรูปที่ 6 เวลารันโปรแกรม THAI.EXE จะต้องมีไฟล์ *.BGI, THAIFONT.CHR และ THAIIFNT.CHR อยู่ในไดรฟ์ที่กำลังทำงานเสมอ

บทสรุป

บทความนี้เป็นแนวทางหนึ่งในการนำภาษาไทยมาแสดงผลจอภาพโดยไม่ต้องพึ่งฮาร์ดแวร์และตัวโปรแกรมที่ใช้มีความอ่อนตัว เพราะใช้ภาษาซี หวังว่าบทความนี้คงจะเป็นหนทางหนึ่งที่จะพัฒนาโปรแกรมภาษาไทยขึ้นมาใช้งานเอง โดยไม่ต้องพึ่งพาโปรแกรมจากต่างประเทศเลย สำหรับโปรแกรมต่าง ๆ ผู้เขียนได้จัดทำเป็นแผ่นซอฟต์เมล์เพื่อบริการผู้ที่สนใจครับ!